VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Plagiarism Detection by Citation Pattern Analysis

Detekce plagiarismu pomocí analýzy citačních vzorů

Jan Sedloň

Bachelor Thesis

Supervisor: doc. Ing. Pavel Krömer, Ph.D.

Ostrava, 2021

Abstrakt

Práce se zabývá detekcí plagiátů pomocí metody známé jako "analýza citačních vzorců". Cílem této práce je popsat již existující postupy detekce plagiátů, najít či vytvořit dataset, který je vhodný na otestování a následně v praxi ukázat algoritmy pro detekci plagiátů pomocí analýzy citací. První část popisuje úvod, terminologii a různé metody detekce plagiarismu. Druhá část se zaměřuje na proces hledání či vytváření vhodného datasetu a třetí čast na konkrétní implementaci a ukázku detekce

Klíčová slova

plagiarismus; algoritmy; detekce; analýza citačních vzorců

Abstract

This bachelor thesis deals with the plagiarism detection method, also known as "citation pattern analysis." This thesis aims to describe existing approaches for detecting plagiarism, find or create a suitable dataset for testing and then show a real life example of citation based plagiarism detection algorithms. First part describes an introduction, terminology and various methods for detecting plagiarism. Second part focuses on the process of finding or creating appropriate dataset and the third part on concrete implementation and show-of of results.

Keywords

plagiarism; algorithms; detection; citation pattern analysis

Acknowledgement

Throughout the writing of this thesis, I have received a great deal of support and assistance. I wish to express my sincere thanks to doc. Ing. Pavel Krömer, Ph.D., for providing me with all the necessary facilities for the research. I also would like to thank Mr. Norman Meuschke for providing me his dataset, which was very much helpful and helping me understand it.

Contents

List of symbols and abbreviations

XML	–	Extensible Markup Language
PDF	–	Portable Document Format
MD5	–	Message-digest algorithm
OCR	–	Optical Character Recognition
AI	–	Artificial Intelligence
PMC	–	PubMed Central
API	–	Application Programming Interface
CITREC	–	Open Evaluation Framework for Citation-based and Text-based Similarity Measures
PDS	–	Plagiarism detection system
BC	–	Bibliographic Coupling
BCS	–	Bibliographic Coupling Strength
LCCS	–	Longest Common Citation Sequence
LCCSS	–	Longest Common Citation Sequence Strength
GCT	–	Greedy Citation Tiling

List of Figures

List of Tables

Chapter 1

Introduction

The Oxford Dictionary defines plagiarism as "*Plagiarism describes the appropriation of other people's ideas, intellectual or creative work and passing them off as one's own .*" [1]. In other words, we can state that plagiarism is copying thought, text, idea, or work without mentioning the original source.

Plagiarism is a major problem of modern times. Both from the point of view of theft of another's work as well as plagiarism detection. The most significant form of plagiarism can be found in the academic sphere. In earlier times, this was not such a problem, as there was no access to a large amount of information. According to Donald McCabe, who researched more than 70,000 students in the United States and Canada, 33% of undergraduate students admitted to cheating on the test in some way [2]. More and more politicians and dignitaries are either accused or convicted of plagiarized bachelor's or master's theses every year. As an example, we can mention the current Prime Minister of the Slovak Republic, Igor Matovič [3][4].

Plagiarism, however, is certainly not the prerogative of academia alone. We can find it almost everywhere. Whether it is the Czech media, in which this frequently appears (the author reads the article on a foreign server, translates it freely, and publishes it without mentioning the source), electronically published photos, or stolen source codes of applications.

In this thesis, we focus mainly on detecting plagiarism by using citation pattern analysis to analyze mainly the academic sphere. So what can be the motivation of a student or academic to plagiarize? The first idea that may come to mind is the availability of information. Some sites contain a huge amount of information, such as Wikipedia, search engines in general, or even ResearchGate, which contains over 135 million professional publications. Such accessibility tempts the person concerned to find what he needs in the shortest possible time. Another aspect that can help is the fact that modern times require more and more demands on the complexity of study as such. The total amount of information available to humanity doubles every 13 months [5]. Therefore, the combination of strong market competition and the amount of information that students must learn creates pressure to find abbreviations. The last aspect that could help is that

students' view of plagiarism is generally that they do not see it as a problem and take plagiarism as a tool to facilitate their studies and move on.

Often overlooked is self-plagiarism. Author cannot use his own words without mentioning the source of which he is author of. The papers usually have a publisher by then and that is a breaking of copyright. It usually is not as serious as other types of plagiarism but the author can be expelled. From the publisher perspective, if author's work is too similar to another, publisher can refuse to publish it [6].

The goals of this thesis are to find existing methods of citation based plagiarism detection, if any exist, create or find a dataset and demonstrate how well these algorithms perform. This does not include parsing citations however. Tests should performed on top of already processed citations, in, for example, a database.

Chapter 2

Types of plagiarism

There are many types of plagiarism. It always depends on the type of document or work and also on what knowledge the author has of plagiarism. If it is someone who is familiar with this topic, they will certainly focus on other methods than someone who simply uses copy-paste. In this chapter, we will look at the basic methods.

2.1 Copy-paste

Copy-paste is the most common type of plagiarism. In this case, the author does not even try to disguise this fact. He finds the thing he needs and copies it to his own work [7]. However, beware, plagiarism is only when copying is prohibited (whether by the nature of its work or by the resource's license terms and intellectual property). Copying the entire text has the advantage of being the fastest solution that requires no extra work. On the contrary, the disadvantage is that it is straightforward to be detected. Both from the point of view of manual examination and from the automated process's point of view. This type of plagiarism should be detectable by virtually any "string-based system."

2.2 Find-and-replace

Another type of plagiarism is changing the order of paragraphs or swapping words. This method is slower and requires manual work. It may confuse some unrefined programs, but most methods are well prepared for this [8].

2.3 Back-translate

One of the very reliable methods is the Back-Translate method. This method works on the principle of obtaining work in a certain language, its free translation into a foreign language, and subsequent

translation back. This almost guarantees that the resulting text will look different from the original[9]. However, this is already very time-consuming, and the only thing the student will help himself with is that he does not have to think about his own topic. The disadvantage arises if it is an academic document. In this case, if it follows the rules of citations, such a document can be detected by analyzing the citation patterns. Even if he changes the whole text, the citations and paraphrases remain, so it is easy to detect.

2.4 Other methods

From some less serious forms, we can mention the following:

- Remix - Paraphrasing the text from various sources so that it looks like the text is complete and these are the author's thoughts [7]
- Recycle - Using your own texts without quoting the original source. It is also called "self plagiarize." [7]
- 404 Error - Text that contains a citation, but the citation refers to a non-existent source. It can be verified using tools such as web.archive.org [7]

Chapter 3

Plagiarism detection

Plagiarism detection is the process of trying to find out whether two independent documents contain identical parts to such an extent that we are able to decide whether one work is plagiarized by the other. The methods can be divided into two parts, ie manual and automatic. Manual inspection of documents is extremely time consuming, especially in the case of a large number of documents. The natural application can be for example checking homework. Because this method requires great memory, constant attention and a large amount of time, we try to find computer methods that would help us with it [10].

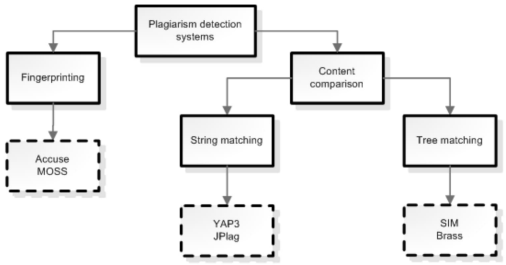

Figure 3.1: Plagiarism detection methods [11]

3.1 Manual

Manual inspection is one way to check if someone has copied their work or part of it. This method is very laborious and is only suitable in particular cases. As a model situation, we can use, for example, a small number of homework, source codes, or essays. In this case, we rely on copying directly between students. We are not able to check (mostly) whether the texts have been copied from the Internet. The advantage of manual control is human deduction and intuition. The teacher can assess whether there are certain similarities in the texts. It can be purely a text or an idea.

3.2 Fingerprint-Based systems

The goal of a fingerprint-based system is to make a fingerprints for all documents in the collection. Fingerprint is a sequence of bytes representing a longer file [11][12]. The simplest form of fast comparison of two files is to convert both to MD5 hash and compare them. Two identical files should result in the same hash. However, for plagiarism detection, this method has to be more sophisticated. This approach can be split into numerous factors like words per line, a number of unique words, etc. The resulting fingerprint can then be compared to another fingerprint. The advantage is obvious, speed. Comparing or calculating a distance between two hashes is considerably faster than using a complex algorithm to find similarities using string or tree matching.

3.3 Content Comparison

3.3.1 String matching

String matching is a different method of detecting plagiarism. Because of its nature, it's much more resource-intensive than fingerprint-based systems. The basic idea is to find identical strings in two documents [13]. Early introduced algorithms were based on Levenshtein distance, which calculates a number of how one string differs from another. One of the most popular string matching algorithms is Running-Karp-Rabin Greedy-String-Tiling [14].

3.3.2 Parse Trees Comparison

Parsing a file into a tree structure is a more advanced variant to string matching. The disadvantage is that it's dependent on a specific file type [15]. Parse tree of a source code file requires to have a different structure than, for example, a natural language text file [12].

At the moment, it is not known whether parse tree comparison is superior to string matching. Since parse tree comparison is more complex than string matching, more research is required [12].

3.3.3 Citation analysis

Citation analysis is the only one detection method that does not rely on textual similarity. Presence of citations is required for this approach to work. Therefore, it's suitable for academic or scientific documents. It is also one of a few algorithms that are language-independent [12].

3.4 Common plagiarism detection systems

There is already a lot of well known systems that do their job well and are used worldwide. Some of them are paid, some of them are not. They differ in speed, accuracy and price.

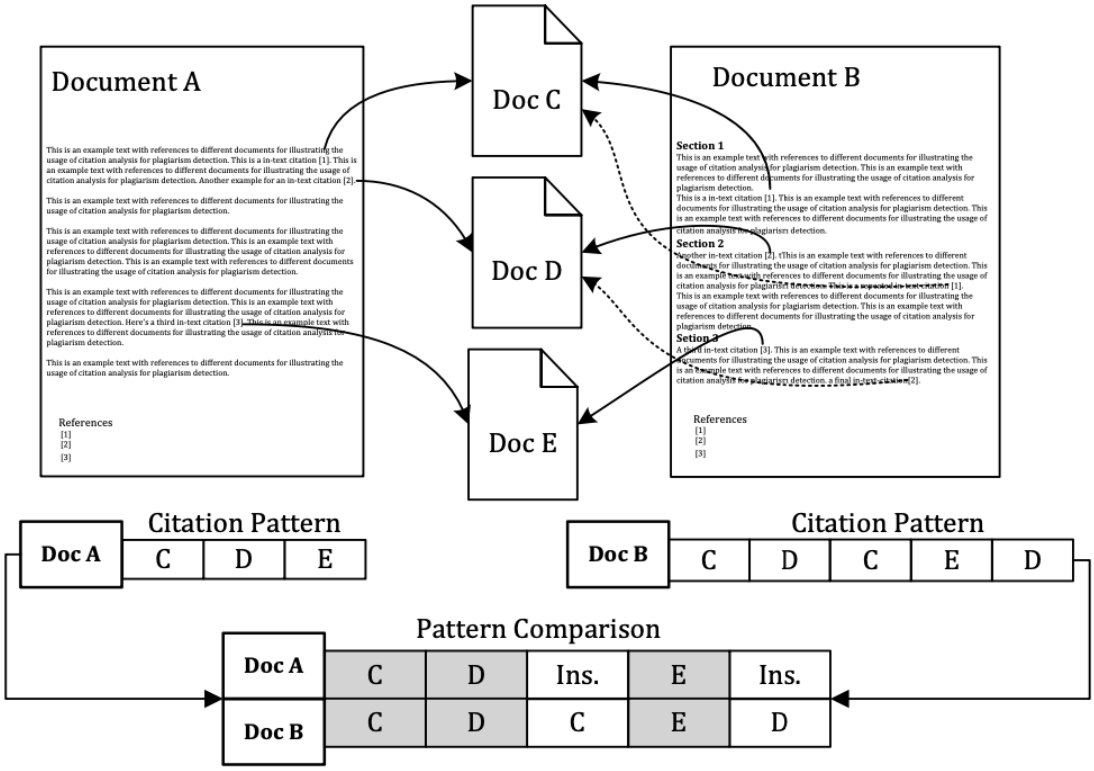

Figure 3.2: Basic algorithm of how citation pattern analysis works [12]

3.4.1 Scribbr

Scribbr is the best and most reliable plagiarism detection tool in the world for 2019. [16] This tool is paid (by the number of words in the document). Still, in return, it can guarantee that your document will not be sold to a third party; you have live support, supports 20 world languages, and have a huge database of websites and documents with which it compares. The fact that they state that they support 20 world languages indicates that their detection method probably does not work based on citation pattern analysis and therefore has no language-independent capabilities. [17]

3.4.2 Unicheck

Unicheck, unlike Scribbr, has a much larger user base. It has over 1 million users and collaborates with more than 1,100 academic institutions around the world. Its advantages include a large document database and also speed. According to the statement on their page, we can get the result as early as 4 seconds after uploading the document. Unicheck is also a paid service. [18]

3.4.3 PlagScan

The third and last tool is the German PlagScan. This tool is very similar to the previous two. It offers integration with educational systems such as Moodle and Canvas, but making your own integration is also possible. Also, it offers a trial version, which previous tools do not provide. [19]

Chapter 4

Dataset for finding plagiarised documents

Dataset is a collection of data made specifically for one or a few specialized use cases. Given the fact that research on how to properly use plagiarism detection using citation pattern analysis is still in its infancy, it is a big challenge to find a suitable dataset. It was necessary to find a dataset that meets several criteria.

1. High probability of finding plagiarism
2. Uniform citation format
3. Availability of a document format suitable for parsing citations
4. Large amount of documents

4.1 arXiv.org

The first source that seemed to be a suitable candidate was arxiv.org.ArXiv is an open dataset of articles and studies from technological disciplines such as physics, mathematics, computer science, and more. The problem with this archive is that most documents are available in PDF format, which makes it very difficult to extract citations and that in this large collection, it would be almost impossible to find similar documents. It would be very complicated to construct a suitable dataset or find documents that they somehow match (they are plagiarised). Another complication was that the citations did not follow a uniform standard. The citation standard problem can be easily solved using a format such as TeX or XML, but most documents were based on PDF. Other formats were random, and we would have to search for which documents have TeX or XML availability manually.

4.2 CITREC and PMC

Dataset was obtained from Dr. Norman Meuschke, who is a colleague of prof. dr. Ing. Bella Gipp, author of the first paper that deals with citation based plagiarism detection. Dataset was manually created in years 2012-2013 and the documents contained in the dataset were by the jury as suspected of plagiarism. Due to this fact, all documents has to be anonymized either by not showing PubMed ID or stripping at least the last two numbers of the id. All documents comes from the open database of PubMed medical papers, or PMC. CITREC is a free open evaluation framework that includes a big database containing documents from PubMed Central Open Access Subset with a various precalculated metrics such as Bibliographic Coupling and preprocessed citations. This dataset is suitable for us mainly because we do not have deal with parsing citations and collecting documents that deal with a similar area [20].

4.3 Extracting citations from PDF documents

4.3.1 arXiv.org

Because the documents from arXiv.org are usually in PDF format and contain different citation standards, I decided not to give up that easily and try to find tools that could make this easier. After some time of searching, I found these promising tools:

- scite.ai
- Grobid
- Adobe Acrobat Pro DC (Trial)

Scite.ai is an excellent tool that can use artificial intelligence to very reliably extract citations from almost any LaTeX or PDF document and even, thanks to a large database of almost 24 million, create a graph of how the citations are interconnected. This tool was not suitable because I did not find any way to export citations to any format.

Grobid is a tool written in the Ruby programming language, which also uses artificial intelligence on learned datasets. Still, after several attempts, I found it unsuitable because it was very unreliable and not only could detect about 50% of citations but those not yet detected completely. On the other hand, the export was in JSON format, which would be relatively suitable for further research.

At Adobe Acrobat Pro DC, I was fascinated by the great OCR at its disposal. We could use this function to extract the document's text into plain text and subsequent processing by a script. I tried to write a Python script that would handle at least 2 citation standards, but again I failed because the standards' rules were much more complex than I originally thought.

After trying all these tools, I gave up the effort and moved to the PMC dataset.

4.3.2 PMC

At first glance, the dataset from PMC seemed a completely suitable candidate. Originally, Norman Meuschke sent an SQL file that contained all the documents suspected of plagiarism. The problem, however, was that it only contained a table with information about the document, such as the ID, file name, and more. The second table already contained the document's text itself, unfortunately without a single citation, so there was no way to process it.

Figure 4.1: SQL Diagram of the first Meuschke's dataset

After further communication, Norman Meuschke provided me with a much larger dataset with a size of 2GB. This dataset was already the closest I needed to the database schema. It already contained fully processed citations, authors, and texts in XML format, but the citations were hard written in the texts as numbers in parentheses. So I tried to write a script that would download the original text directly from PubMed. However, it turned out that most documents have already been deleted or marked as unavailable.

The last source Norman Meuschke provided me was the CITREC project which follows the definition given on the page "*CITREC is an open evaluation framework for citation-based and text-based similarity measures.*" [21] The dataset on this page already contained everything I needed. Metadata about documents from PubMed, authors, citations, and the position of citations in the text. The final SQL file, which I subsequently downloaded and used for further research, contained over 260,000 documents and almost 15,000,000 citations.

To deal with the case where I would need files and texts of documents in the research, I used a text file from the pages FTP documentation PubMed , which contains metadata about all the files on their FTP server and wrote a short Python script that performs a SELECT on the local database and uses FTP to download all the documents I need. Because many documents can be downloaded at the same time, multithreading was used.

Chapter 5

Citation based plagiarism detection - Analysis and Implementation

Detection of plagiarism by citation pattern analysis is still a relatively new and unexplored area. The first academic who came up with the idea to start researching citation patterns is prof. dr. Ing. Bela Gipp, who elaborated this topic in detail in his professional work "Citation-based Plagiarism Detection - Detecting Disguised and Cross-language Plagiarism using Citation Pattern Analysis" [12].

The analysis of citation patterns in the current phase is certainly not the holy grail of all plagiarism detection. However, it is a very suitable complement to existing methods, both because of their speed and their independence from the work's language. Language independence is a huge advantage of this method, as many alternatives do not exist and are therefore particularly suitable for back-translate plagiarism (see Section 2.3).

In this section, we describe what tools were used, what Bibliographic Coupling is and how it relates to CbPD, as well as other citation analyzes that were originally used for purposes other than plagiarism detection. We will also learn about several algorithms that arose from algorithms that were originally used to detect plagiarism on the basis of string-matching, and in the last part we will introduce the subsequent implementation in Python and the results of our research embedded to the Meuschke's dataset.

5.1 Citation based similarity measures

There has been a lot of research on citations and similarities in documents. As early as 1963, M.M introduced a concept known as Bibliographic Coupling [22]. Subsequently, in 1973, Irena Marshakova-Shaikevich introduced Co-citation [23]. However, all of these methods were designed purely for the classification of relatedness and not for similarity.

5.1.1 Bibliographic Coupling

Bibliographic Coupling is one of the oldest and most widespread citation-based methods for calculating relatedness. This method was originally used as a metric that could tell us if two documents address a similar topic.

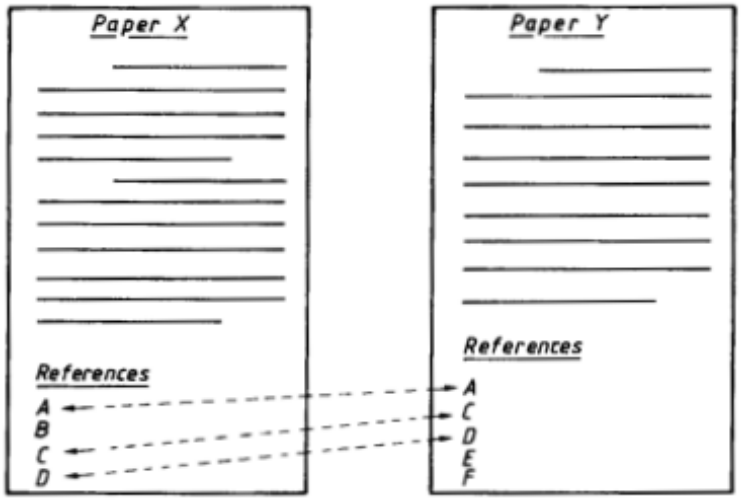

Figure 5.1: Bibliographic coupling strength of three [24]

The higher the number of shared citations, the more related documents are. The mathematical representation of Bibliographic Coupling Strength can be found in Equation 5.1.

$$BCS(d1, d2) = \frac{|R_{d_1} \cap R_{d_2}|}{|R_{d_1} \cup R_{d_2}|} \tag{5.1}$$

The variable d is a document and R_d is a set of documents that are cited by d. The highest possible result is 1, which means that both documents have exactly the same citations [12].

The disadvantage of this method is that it is independent of the order of citations. Thanks to this, we can say that the documents are related, but not whether one was copied from the other.

5.1.2 Co-citation

Unlike Bibliographic Coupling, co-citation does not measure relatedness based on the number of identical citations, but rather the number of documents that cite both documents being compared at the same time. The problem with Co-citation is that the newer the documents, the less cited they are. For the needs of CbPD, this procedure is also not suitable due to the fact that the bachelor's theses are cited absolutely minimally and therefore it would be unsuitable to find whether two documents are related to each other. See Figure 5.2 for an example of two documents with Co-citation strength of three.

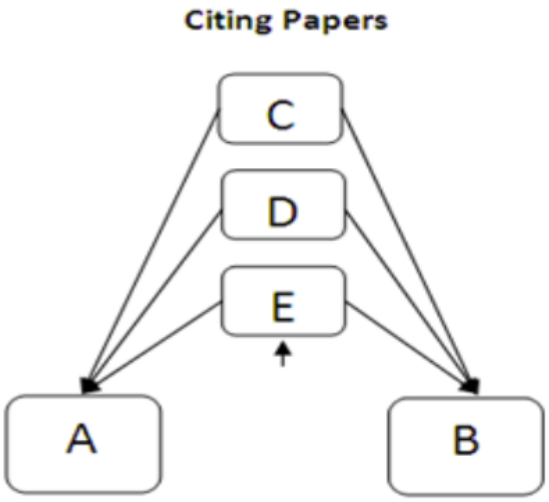

Figure 5.2: Co-citation strength of three. Both documents are cited by other three documents [25]

5.2 Algorithms

When studying the possibilities of using citations in the detection of plagiarism, Bela Gipp found no other research to do anything similar [12]. There were no algorithms to measure the similarity of consecutive citations. We were able to evaluate well whether the two documents are similar, but that is only global similarity. Thus, no such algorithm existed for local similarity. Bela Gipp was thus inspired by existing algorithms used for string similarity, namely Longest Common Subsequence and Greedy String Tiling, and adapted them for use in citations.

5.2.1 Longest Common Citation Sequence (LCCS)

The longest common citation sequence follows the Bibliographic Coupling Strength and Longest Common Subsequence. Because Bibliographic Coupling calculates a global match of citations, LCCS calculates a local match by sticking to a given order. LCCS is therefore a collection of citations that are in consecutive order in both documents [26]. This very simple method has its pitfalls. Suitable only in cases where the original text has been copied and almost unchanged. Once positions of sections and citations are changed, LCCS's success declines rapidly. Below we can find a pseudocode of this algorithm.

Algorithm 1: LCCSS algorithm

```
Result: LCCS
lastOccurenceIndex ← -1;
lccsSet ← empty set;
for each citation in the first document do
    if citation in document2Citations starting at index lastOccurenceIndex + 1 then
        lastOccurenceIndex ← index of found citation in the second document's citations
          starting at index lastOccurenceIndex + 1;
        lccsSet ← lccsSet ∪ {document2Citations[lastOccurenceIndex]};
    end
end
if lengthOf(first document's citations) or lengthOf(second document's citations) or
 lengthOf(lccsSet) then
    return empty set
end
return lccsSet
```

5.2.2 Greedy Citation Tiling

Greedy Citation Tiling looks for the longest possible, sequential sequence of citations, called tiles. The method is especially useful for documents plagiarized using the shake & paste method, which swaps certain sections in a document so that it is not captured, for example, by LCCS. Tiles are arrays of citations that share the same sequence, but regardless of where they occur. Tile can be written as tuple $t = (s_1, s_2, l)$, where s_1 is the initial position of the sequence in the first document, s_2 is the initial sequence in the second document and l is the number of citations in the sequence [27]. We can optionally give GCT a number parameter n to only catch those tiles that are longer than n. See Figure 5.3 for an example of how tiles are made. Algorithm below that is an example how it works in practise.

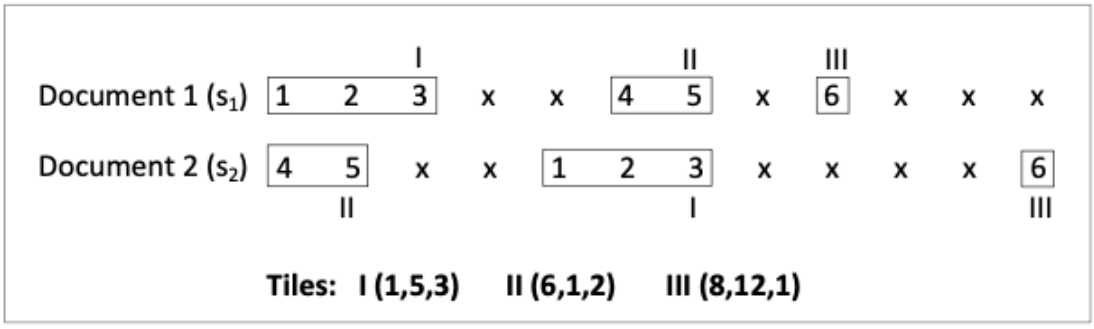

Figure 5.3: Greedy Citation Tiles [12]

```
Result: GCT Strength
tiles ← empty set;
if lengthOf(first document's citations) or lengthOf(second document's citations) or
 lengthOf(sharedDocuments(first document, second document)) then
  return 0
end
for each citation and its index in the first document do
    for each citation and its index in the second document do
        if second document's citation index already in tiles then
            skip iteration;
        end
        if first document's citation equal second document's citation then
            firstIndex ← first document's citation index;
            secondIndex ← second document's citation index;
            tileLength ← 0;
            while true do
                if firstIndex ≥ lengthOf(first document's citations) or secondIndex ≥
                 lengthOf(second document's citations) then
                    end loop;
                else if first document's citation on index firstIndex equals second document's
                 citation on index secondIndex then
                    firstIndex ← firstIndex + 1;
                    secondIndex ← secondIndex + 1;
                    tileLength ← tileLength + 1;
                else
                    break loop;
                end
            end
            if tileLength ≥ minimal length of a single tile then
                tiles ← tiles ∪ {tupleof firstdocument′scitationindex +
                  1, seconddocument′scitationindex + 1, tileLength};
            end
        end
    end
end
return tiles
```

5.3 Used tools and services

5.3.1 Python

Python is an interpreted scripting programming language that was created in 1991 by Guido van Rossum [28][29]. The language excels mainly because spaces are used as block separators instead of notoriously known curly braces. We can choose whether we prefer two or four spaces. It always depends on the coding style guide that we stick to [30][31]. I chose this language because it is a relatively simple and clear language. In newer versions, it already has excellent typing support for much better code completion, and, unlike Javascript, it supports types natively. In the case of Javascript, we would have to use either the Typescript transpiler or the JSDoc documentation syntax, which, however, can be unnecessarily complex very quickly. The last reason is that Python holds a major position in the data science usability rankings [32][33]. For this bachelor thesis, I decided to use the newest version, Python 3.9.

5.3.2 PyCharm

PyCharm is an IDE created by the Czech company JetBrains [34], which already has many IDEs for various languages. PyCharm provides first-class support for Python, an integrated tool for working with various database systems and tools directly for Data Science.

5.3.3 DataGrip

DataGrip is another software from JetBrains, but this time specifically for database systems. Unlike integrated database support with the PyCharm plugin, I decided to use this because it is much more convenient when working with many tables.

5.3.4 Redis

Redis an in-memory storage [35], that is often use as a cache layer between layers. It plays a minor role in this research. Redis is useful if we run database queries that take a very long time to complete and we simultaneously develop an application. Each time we run the application, Redis can save us several minutes before the query is executed.

5.3.5 CITREC

4.3.2 CITREC is an open evaluation framework for citation-based and text-based similarity measures [21]. I used this framework provided by Norman Meuschke because of its large dataset and because many metrics are already pre-calculated, which not only greatly simplifies the work but also helps with the subsequent check if my subsequent plagiarism detection works as it should.

5.4 Terminology

5.4.1 Citation vs reference

Common misconception is that in the vast majority of cases, citations and references have a different meaning than originally proposed. If we have a document A, that paraphrases part of the document B, then the document A contains a reference to the document B and the document B gets a citation from the document A [24]. Nowadays, however, the words reference and citation ares used interchangeably. To avoid confusion, we will use the words citations and references in the same meaning, ie. in-text citation.

5.5 Implementation

5.5.1 Final dataset

The resulting dataset used is a combination of a PubMed dataset with a dataset provided by Norman Meuschke, which contains pairs of documents that were marked by the jury as pairs that are demonstrably similar, but were not marked as plagiarism and a factional one custom made specifically for this test. The dataset obtained from Normal Meuschke contains a total of four groups of ten pairs of documents. Groups differ in the method of plagiarism involved. Custom made one contain 6 pairs of documents in total, grouped by 3 pairs. First group was made with Copy & Paste method in mind and the second one with Shake & Paste. The custom dataset can be seen in detail in Figure 5.1. Numbers in *Citations* represents shared citations that two documents have in common. *id* is a fictional PubMed document ID. To show a possible inaccuracy of LCCSS and GCTS, small BCS flaw was applied on the third pair. Scaling is an operation in which author places not shared citations to break the consequent order and therefore lowers GCTS.

In some cases, the Meuschke's dataset lacks citation information for suspected documents. Closer inspection discovered although the sources of the citations were the same, some lacked an author which was set to *NULL*. Therefore, the primary condition is that name of the cited document must be equal for both citations. In a case where one of the authors is missing, string matching of authors is skipped. On the other hand, if both records have an author not empty, string matching takes authors in the account as well. Sometimes, documents has duplicated citations and therefore results can vary. This is eliminated by the custom dataset.

ID	Common citations													Purposeful edit
10000000	1	x	x	2	3	4	x	5	x	x	6	7	8	
10000001	1	x	2	3	4	x	x	5	6	7	8	x	x	
10000002	x	x	1	x	x	2	3	4	5	x	x	6	7	
10000003	1	2	x	x	3	4	5	x	x	6	7	x	x	
10000004	1	2	3											Small BCS
10000005	x	x	x	x	1	2	x	x	x	x	x	x	3	
10000006	1	x	2	x	3	4	5	x	6	x	7	8	9	
10000007	3	4	5	x	x	x	1	x	6	x	7	8	9	
10000008	x	x	1	x	2	3	x	4	5	x	x	6	7	
10000009	4	5	x	x	1	2	3	x	x	6	7	x	x	
10000010	x	1	2	3	4	x	x	5	x	x	6	7	x	Scaling
10000011	3	4	x	5	x	1	2	x	x	6	x	7	x	

Table 5.1: Custom dataset

5.5.2 Calculating strength for algorithms

Since both all algorithms, LCCS and BC return some sort of a collection, we need a number that represents strength of the results. Each result can be anything between 0 and 1, with 1 being identical.

5.5.2.1 Longest Common Citation Sequence Strength

Result of Longest Common Citation Sequence is an array containing citations that follow identical order of the compared document. To calculate it, we take the length of the resulting array and divide it by the number of shared citations. We can see it in detail in Equation 5.2.

$$LCCSS(d_1, d_2) = \frac{|LCCS(d_1, d_2)|}{|s_1|} \tag{5.2}$$

If $LCCS(d_1, d_2)$ is a function that returns a set of shared citations that follow the same order and s_1 being a set of all shared citations that first document have in common with the second one, both values are divided and the maximum number can be 1, which means that all shared citations follow the same order.

5.5.3 Greedy Citation Sequence Strength

Greedy Citation Sequence Strength goes with the same idea as Longest Common Citation Sequence Strength. We take the length of all tiles and we divide by the number of shared citations.

$$GCTS(d_1, d_2) = \frac{\sum\limits_{x \in t} x_2}{|s|} \tag{5.3}$$

If t is a collection of tuples (tiles) whose third element is a length of the tile, we add it up and divide it by the number of shared citations.

5.5.4 Discussion

Beware, LCCSS and GCTS results are heavily biased by and dependent on shared citations. If two documents have really low BCS, the likeliness of high LCCSS and GCTS increases rapidly. For example if two documents share only two citations out of 150 in total, the chance of being in the same order is 50% and then the LCCSS would be 1 even though they are not similar at all. The key for an accurate result is to choose right coefficient of BCS.

5.6 Results

In Figure 5.4 we can see the results of applied algorithms. The $0 - 9$ pairs belong to the Copy & Paste group, which we can also see in the results. The first ten pairs have a very high LCCSS

index, often approaching 1. In the second group $10-19$ belonging to the Shake & Paste group, the values are often already lower, but not so noticeably. This may be due to the relatively low BCS, which increases the probability of the same sequences, see Section 5.5.4. As for the GCT algorithm, it performs very well in all groups. The minimum number of matching citations in a sequence has been set to 2. The difference compared to LCS can be seen mainly in the second group, where it was able to detect groups of citations that were shuffled.

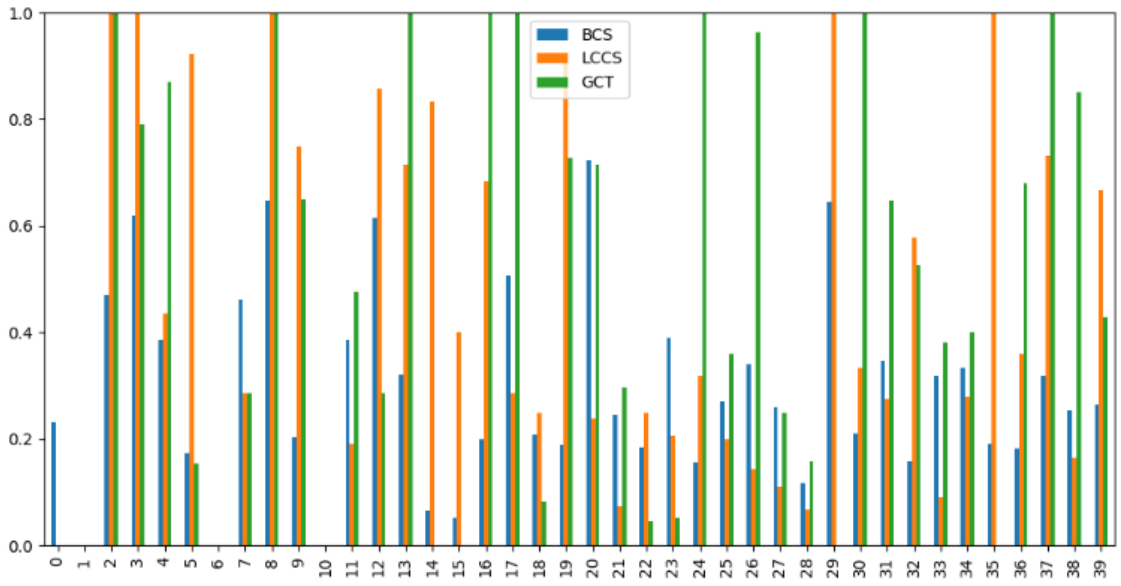

Figure 5.4: LCCS and BCS results on Meuschke's dataset

In Figure 5.5 are results that come from our own dataset. Even though this dataset is very small and fictitious, it was created directly for the purpose of testing plagiarized documents using Copy & Paste a Shake & Paste. The dataset contains six pairs of documents. The first three using Copy & Paste and the second three using Shake & Paste. According to the table 5.1, an adjustment was made to two pairs, which can outline some problems.

As we can see, the first two pairs have a value of 1 for LCCSS. This was expected as all common citations are in the same order. GCTS also has a high value, but is not equal to 1 because the minimum tile length is set to 2 quote. This means that even if there are two identical, isolated quotations in the text, we will not take them into account.

In the third pair, where the first document has only 3 citations and they are all common to the second document, we can see a misleading result. As I mentioned in the 5.5.4 section, a very small BCS value can lead to skewed LCCSS and GCTS results, as the lengths of the individual fields are divided by the length of the field containing the common citations.

From the second half of the dataset to which Shake & Paste was applied, we find that the LCCSS values are slightly lower than in the previous case. This is due to the fact that the citations

are shuffled. A much bigger problem would arise if the first citation of the first document were the last citation of the second document. Then the LCCSS would be almost zero. GCTS is still doing very well, as it does not depend on the order of citations. In the last pair we can see a significant decrease in LCCS. This is because the so-called Scaling was applied to one of the documents, or in other words, the insertion of different citations between a group of common citations. GCT is then unable to recognize a longer set of citations.

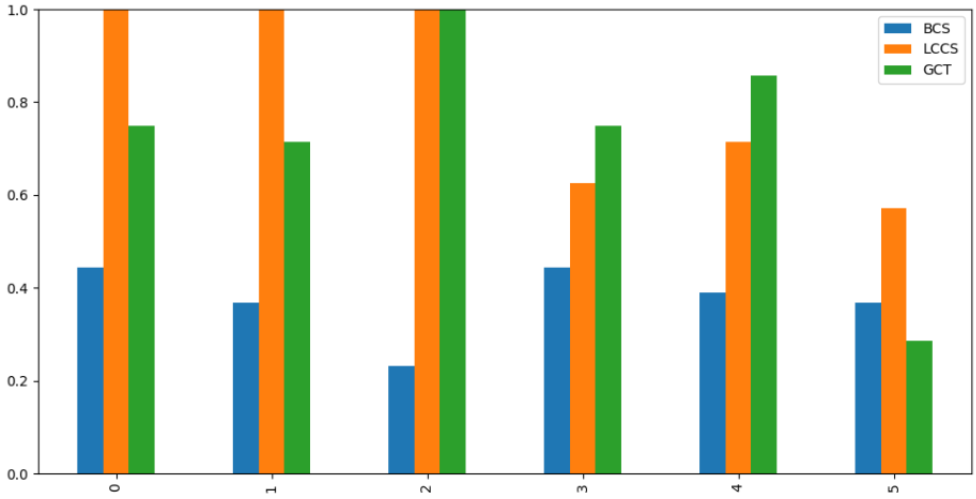

Figure 5.5: LCCS and BCS results on the new dataset

5.7 Observation

From the results of the datasets, we found that with sufficient information about citations, these algorithms and CbPD approach in general have great potential overall. LCCS is suitable for Copy & Paste plagiarism and GCT for Shake & Paste. There are many types of plagiarism and these algorithms are not suitable for all of them. It is necessary to take into account the high values of LCCSS and GCTS in the case of low BCS, or alternatively to devise a way to reduce these values based on BCS.

Chapter 6

Conclusion

The goal of this thesis was to find if existing citation based plagiarism detection methods exist and whether they are usable in the real world. The results were satisfactory. Due to the nature CbPD, its biggest advantage is that for the currently available algorithms, text of the documents is not needed needed and therefore, it is language-independent, which is a big advantage in comparison to other algorithms.

There is no program that could tell us if one document is plagiarized from another. This is why we need to developer more and more complicated algorithms and ways to detect it. Citation based plagiarism detection method is one of many. It certainly is not meant to be used as the only one method for every use case. However, it is a good method that can complement others, especially in academic area or any area whose documents include citations.

In this thesis I described types of plagiarism, how manual control and content based methods work and then in detail researched how citation based approach can be used. Two datasets were used to partially eliminate a chance of faulty results.

During this thesis I learnt a lot about introduction to data analysis, its usages and also how widespread plagiarism is.

6.1 Future work

Throughout writing this thesis, I came across a few issues and optimizations that could be thought through in a future research.

6.1.1 Root citation

The fact that a citation appears in the text does not necessarily mean that this particular part of the cited document is the original citation. By that, I mean that if we quote or paraphrase a part of a document, that part in that document can be quoted from elsewhere. Let's have a simple example.

Let's have a document d_1 a document d_2. Document d_1 contains an information i_1 from Wikipedia. Document d_2 contains information i_1 from document d_3. If we take a closer look, information i_1 was originally taken from document d_1. This way, the citations are different and were not detected by CbPD systems. Although this is quite debatable, this may indicate both that the documents were not really plagiarized and that the author has done his job so that it does not look like plagiarism and has found the very first source of the text.

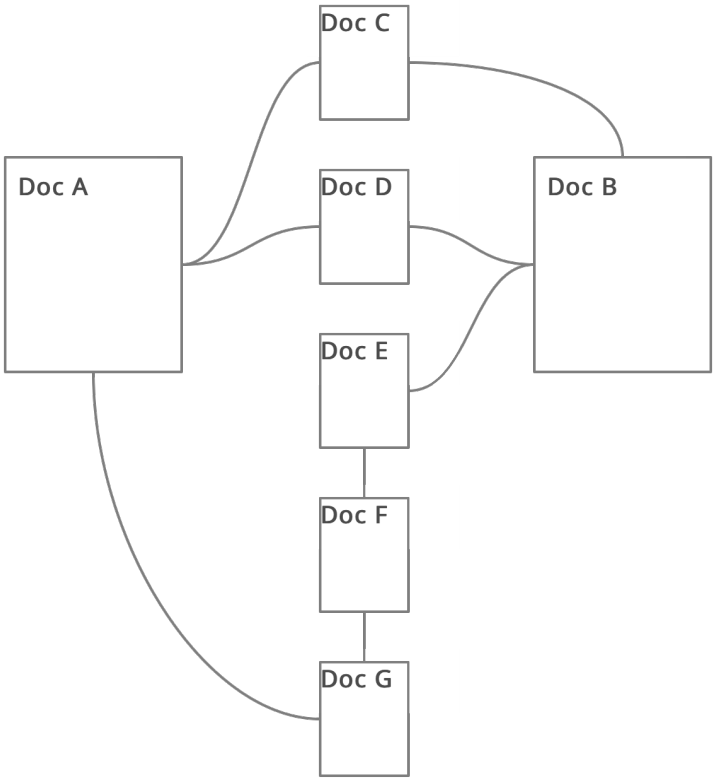

Figure 6.1: Example of root citation usage

6.1.2 Streaming

Most PDSs work by giving them some work, and it can take days, for example, for the final result to appear. If we want to upload a large number of documents, depending on the available computing power, it can take much longer. The optimization idea that came to my mind comes from the well-known NP-hard problem of a business traveler and its optimization using ant colony algorithms.

If we analyze it, the problem of a business traveler is that we have a specified n number of points/cities that the business traveler wants to go through and return, and the task is to find a combination that will guarantee that he will really go the shortest way. The naive implementation has the complexity $O(n!)$. But do we need a 100% correct result at the cost that its computation time can easily exceed the human race's existence? Can't we get a result with 95% accuracy in the time we are willing to wait? We can. We can take inspiration from the optimization of ant colonies, in which the result is shown immediately and over time improves when more and more ants improve their path and look for a better result. To detect plagiarism, suspicion is ample for us at the beginning. Currently, technology cannot tell us reliably and unambiguously whether a document is plagiarized or not (in most cases). In order for the result to be as reliable as possible, either very performance-intensive algorithms must be used, or the algorithms must be combined, which also leads to an increase in inspection time. For example, if we use percentages as units, like 60% or 70 % is enough to give us a feeling that something is not right.

6.1.3 Intentional root citation

As I have already outlined the problem, which I called *Root Citation*, it can actually be used to improve the algorithm. If documents d_1 and document d_2 share multiple citations from document d_3, ie. $BibliographicCouplingStrength(d_1, d_2) > 0.5$, for each citation i in a similar position we can find out whether the text located in d_3 is the original text or text again quoted from elsewhere. In this way, we can go through the citations to the original text and compare the citations with each iteration. The big disadvantage of this solution is that we would need a huge database of texts, pages, and documents to browse citations. Many cited websites may no longer exist, documents may not exist in digital form, and there may not be a way to process them for other cases.

Bibliography

1. *Concise Oxford Companion to the English Language* [online]. Oxford University Press, 2003-01-01 [visited on 2021-03-14]. ISBN 9780192800619. Available from DOI: 10.1093/acref/9780192800619.001.0001.
2. MCCABE, Donald L. Cheating among college and university students: A North American perspective. *International Journal for Educational Integrity* [online]. 2005-11-29, vol. 1, no. 1, p. 4 [visited on 2021-03-09]. ISSN 1833-2595. Available from DOI: 10.21913/IJEI.v1i1.14.
3. HAVLICKÁ, Kateřina. Slovenský premiér Matovič také opisoval, je stejný plagiátor jako Danko [online]. [N.d.] [visited on 2021-03-09]. Available from: https://www.idnes.cz/zpravy/zahranicni/igor-matovic-slovensko-premier-plagiat-diplomova-prace.A200716_142204_zahranicni_kha.
4. MAS. Matovič opisoval v diplomové práci, píše Denník N. „Nikdy jsem se svým titulem nechlubil," reagoval premiér [online]. [N.d.] [visited on 2021-04-11]. Available from: https://ct24.ceskatelevize.cz/svet/3141887-matovic-opisoval-v-diplomove-praci-pise-dennik-n-nikdy-jsem-se-svym-titulem-nechlubil.
5. SCHILLING, David Russell. Knowledge Doubling Every 12 Months, Soon to be Every 12 Hours [online]. [N.d.] [visited on 2021-03-09]. Available from: https://www.industrytap.com/knowledge-doubling-every-12-months-soon-to-be-every-12-hours/3950.
6. STREEFKERK, Raimo. *What is self-plagiarism and how can you avoid it?* [Online] [visited on 2021-04-22]. Available from: https://www.scribbr.com/plagiarism/self-plagiarism/.
7. R, SOMASUNDARAM. *10 Types of Plagiarism – Every Academic Writer Should Know* [online] [visited on 2021-03-14]. Available from: https://www.ilovephd.com/10-types-of-plagiarism-every-academic-writer-should-know/.
8. ROKA, Yam Bahadur. Plagiarism: Types, Causes and How to Avoid This Worldwide Problem. *Nepal Journal of Neuroscience* [online]. 2017-12-01, vol. 14, no. 3, p. 4 [visited on 2021-04-22]. ISSN 1813-1956. Available from DOI: 10.3126/njn.v14i3.20517.

9. JONES, Michael. Back-translation: the latest form of plagiarism [online]. [N.d.], pp. 4–5 [visited on 2021-04-22]. Available from: https://ro.uow.edu.au/cgi/viewcontent.cgi?article=1014%5C&context=apcei.

10. BRETAG, Tracey; MAHMUD, Saadia. A model for determining student plagiarism: Electronic detection and academic judgement. *Journal of University Teaching & Learning Practice* [online]. 2009, vol. 2009, no. 1 [visited on 2021-04-23]. Available from: https://ro.uow.edu.au/cgi/viewcontent.cgi?article=1076%5C&context=jutlp.

11. MOZGOVOY, MAXIM. *ENHANCING COMPUTER-AIDED PLAGIARISM DETECTION*. University of Joensuu, Computer Science and Statistics, 2007. ISBN 978-952-219-049-9. Disertace. University of Joensuu.

12. GIPP, Bela. *Citation-based Plagiarism Detection*. Wiesbaden: Springer Fachmedien Wiesbaden, 2014. ISBN 978-3-658-06393-1. Available from DOI: 10.1007/978-3-658-06394-8.

13. PARWITA, Wayan Gede Suka; INDRADEWI, I Gusti Ayu Agung Diatri; WIJAYA, I Nyoman Saputra Wahyu. String Matching based Plagiarism Detection for Document in Bahasa Indonesia. In: *2019 5th International Conference on New Media Studies (CONMEDIA)*. 2019, pp. 54–58. Available from DOI: 10.1109/CONMEDIA46929.2019.8981821.

14. WISE, Michael. String Similarity via Greedy String Tiling and Running KarpRabin Matching. *Unpublished Basser Department of Computer Science Report*. 1993-01.

15. SONG, Hyun-Je; PARK, Seong-Bae; PARK, Se. Computation of Program Source Code Similarity by Composition of Parse Tree and Call Graph. *Mathematical Problems in Engineering*. 2015-04, vol. 2015, pp. 1–12. Available from DOI: 10.1155/2015/429807.

16. STREEFKERK, Raimo. *The best plagiarism checkers of 2019* [online] [visited on 2021-04-05]. Available from: https://www.scribbr.com/plagiarism/best-plagiarism-checker/.

17. *Scribbr* [online] [visited on 2021-04-11]. Available from: https://www.scribbr.com.

18. *Unicheck* [online] [visited on 2021-04-11]. Available from: https://unicheck.com.

19. *PlagScan* [online] [visited on 2021-04-11]. Available from: https://www.plagscan.com/en/.

20. GIPP, Bela; MEUSCHKE, Norman; LIPINSKI, Mario. CITREC: An Evaluation Framework for Citation-Based Similarity Measures based on TREC Genomics and PubMed Central. In: *Proceedings of the iConference 2015*. Newport Beach, California, 2015-3 24-27. Available also from: http://ischools.org/the-iconference/.

21. *CITREC* [online] [visited on 2021-04-02]. Available from: https://dke.uni-wuppertal.de/en/projects/citrec.html.

22. KESSLER, M. M. Bibliographic coupling between scientific papers. *American Documentation*. 1963, vol. 14, no. 1, pp. 10–25. Available from DOI: https://doi.org/10.1002/asi.5090140103.

23. MARSHAKOVA-SHAIKEVICH, Irena. System of Document Connections Based on References. *Nauchn-Techn.Inform.* 1973, vol. 1973, no. 6, pp. 3–8.

24. EGGHE, Leo; ROUSSEAU, Ronald. Introduction to Informetrics. Quantitative Methods in Library, Documentation and Information Science [online]. 1990, p. 204 [visited on 2021-04-07]. Available from: `https://www.researchgate.net/publication/28803158_Introduction_to_Informetrics_Quantitative_Methods_in_Library_Documentation_and_Information_Science`.

25. SURWASE, Ganesh; SAGAR, Anil; KADEMANI, B. S.; BHANUMURTHY, K. *Co-citation Analysis: An Overview*. Scientific Information Resource Division, Bhabha Atomic Research Centre, Trombay, Mumbai (India), 2011. ISBN 935050007-8.

26. GIPP, Bela; MEUSCHKE, Norman. Citation pattern matching algorithms for citation-based plagiarism detection. In: *Proceedings of the 11th ACM symposium on Document engineering - DocEng '11* [online]. New York, New York, USA: ACM Press, 2011, pp. 254–255 [visited on 2021-04-22]. ISBN 9781450308632. Available from DOI: `10.1145/2034691.2034741`.

27. GIPP, Bela; MEUSCHKE, Norman; BREITINGER, Corinna; PITMAN, Jim; NÜRNBERGER, Andreas. Web-based Demonstration of Semantic Similarity Detection using Citation Pattern Visualization for a Cross Language Plagiarism Case. In: 2014-04, vol. 2.

28. *The Making of Python. A Conversation with Guido van Rossum, Part I* [online]. 2003- [visited on 2021-04-11]. Available from: `https://www.artima.com/articles/the-making-of-python`.

29. MCKINNEY, Wes. *Python for data analysis: data wrangling with Pandas, NumPy, and IPython.* Second edition. Sebastopol: O'Reilly, [2018]. ISBN 9781491957660.

30. *PEP 8 – Style Guide for Python Code* [online] [visited on 2021-04-05]. Available from: `https://www.python.org/dev/peps/pep-0008/%5C#indentation`.

31. *Google Python Style Guide* [online] [visited on 2021-04-05]. Available from: `https://google.github.io/styleguide/pyguide.html%5C#34-indentation`.

32. *TOP 10 DATA SCIENCE PROGRAMMING LANGUAGES FOR 2020* [online] [visited on 2021-04-05]. Available from: `https://www.analyticsinsight.net/top-10-data-science-programming-languages-for-2020/`.

33. *Top Programming Languages for Data Science in 2020* [online] [visited on 2021-04-05]. Available from: `https://towardsdatascience.com/top-programming-languages-for-data-science-in-2020-3425d756e2a7`.

34. *JetBrains Corporate overview* [online] [visited on 2021-04-12]. Available from: `https://resources.jetbrains.com/storage/products/jetbrains/docs/corporate-overview/en-us/jetbrains_corporate_overview.pdf`.

35. *Redis* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [visited on 2021-04-08]. Available from: `https://en.wikipedia.org/wiki/Redis`.